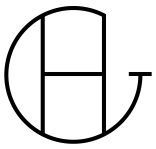


HoGent

Faculteit Bedrijf en Organisatie

Proactief monitoren aan de hand van log files en Elastic stack

Bert Vervaele

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Bert Van Vreckem
Co-promotor:
Martin Van Den Abeele

Instelling: Oxya

Academiejaar: 2016-2017

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Proactief monitoren aan de hand van log files en Elastic stack

Bert Vervaele

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Bert Van Vreckem
Co-promotor:
Martin Van Den Abeele

Instelling: Oxya

Academiejaar: 2016-2017

Tweede examenperiode

Samenvatting

De analyse van log files is een onderwerp dat recent aan belang gewonnen heeft en oXya,de opdrachtgever achter de bachelorproef, had dit dan ook graag in meer detail bekeken. Hier zijn enkele tools voor te vinden, maar de Elastic stack trok hun aandacht. Dit omwille van de schaalbaarheid met grote hoeveelheden data, maar ook omdat het open source is en het alles lijkt te omvatten wat voor hen gewenst is. De opzet van deze bachelorproef is om te onderzoeken wat de mogelijkheden met Elastic stack zijn omtrent proactief monitoring aan de hand van log files.

Na een requiements analyse werden hun vereisten neergeschreven in het hoofdstuk casus. Er werd onderzocht in welke mate Elastic stack aan hun vereisten voldeed door het ïmplementeren van een proof of concept. De proof of concept werd uitgewerkt voor twee besturingssystemen en twee databanken.

Dit werd uitgewerkt voor de drie core elementen en vier extra tools. In Logstash werd de input bekeken en hoe deze genormaliseerd kan worden. Eveneens werd er aandacht besteed aan het schrijven van een config file. Voor Elasticsearch wordt een beeld geschetst van hoe de databank juist werkt en hoe een zoekopdracht uit te voeren. Elasticsearch maakt het mogelijk om enkele tools erg efficiënt te laten werken. Deze tools bevatten de functionaliteiten die nodig waren voor deze casus. In de grafische interface Kibana kunnen dan beduidende grafieken gemaakt worden. Om sommige vereisten aan te tonen waren ook theoretische berekeningen nodig.

Elastic stack voldeed aan elke voorwaarde, al is verder onderzoek naar een optimale anomalie detectie methode wel aangeraden. Er kwamen reeds enkele mogelijkheden aanbod, maar deze zijn nog voor verbetering vatbaar. Elastic stack is een snel groeiend product en zal in de toekomst alleen maar meer mogelijkheden omvatten.

Voorwoord

Deze is bachelorproef geschreven binnen het kader van het verplicht opleidingsonderdeel Bachelorproef voor het behalen van een diploma bachelor in de Toegepaste Informatica.

Stageplaats

Het onderwerp werd voorgesteld door mijn stagebedrijf. Daarom daarvoor zou ik graag mijn stageplaats oXya willen bedanken. Niet alleen voor het onderwerp maar ook de tijd die ze besteed hebben om mij te helpen in dit onderzoek.

- Dhr. Martin Van Den Abeele (co-promotor) - Bij hem kon ik altijd terecht met mijn technische vragen en hij bezorgde mij nuttige data. Dit zorgde voor een prettige samenwerking in een zeer leerrijk proces.
- Dhr. Lukas Becue - Hij ielp met het onderhouden van de elastic stack en het creëren van testdata.

Familie

Ik zou graag mijn ouders bedanken voor de steun, niet alleen nu maar doorheen heel mijn studies. Ook voor hun bijdrage tot het verbeteren van deze bachelorproef, en verbeteren in de ruime zin van het woord.

Docenten aan de HoGent

- Dhr. Bert Van Vreeckem (promotor) - Hij volgde deze bachelorproef op en verzorgde nuttige feedback die leidde tot het eindresultaat.
- Dhr. Buysse Jens - Voor het in goede banen laten lopen van dit opleidingsonderdeel en de goede communicatie er rond.

Inhoudsopgave

1. Inleiding

1.1 De opdrachtgever, Oxya

Oxya is een internationaal bedrijf met onder meer een vestiging in Kortrijk. Het bedrijf houdt zich enkel en alleen bezig met het hosten van SAP-systemen. Sap is een ERP-pakket dat gebruikt wordt voor de ondersteuning van bedrijfsprocessen. Bedrijven vanaf enkele werknemers gebruiken dit voor het bijhouden van financiële gegevens, voorraad aantallen, contactgegevens, ... Als de servers niet naar behoren functioneren kan het voorkomen dat een heel bedrijf stil ligt tot de servers terug normaal functioneren. Het is dus hun taak om te monitoren als alles werkt zoals het hoort. Daarom zijn ze steeds opzoek naar nieuwe manieren om hen hierbij te helpen.

Ze hebben reeds twee tools ter beschikking, namelijk PRTG en een eigen ontwikkelde tool koaly. Beide tools vragen systeemgegeven op en gaan dan kijken als de waarden niet te hoog of te laag zijn. Om te kijken als de waarden niet te hoog of te laag zijn, zijn limieten vastgelegd en deze worden thresholds genoemd. Deze limieten staan vast en moeten dus handmatig gewijzigd worden indien nodig. Dus momenteel maakt oXya enkel gebruik van statische thresholds in hun tools.

1.2 Probleemstelling en Onderzoeksvragen

De vraag voor dit project komt dus vanuit het bedrijf Oxya. De concrete vraag staat beschreven in 3. De opzet van deze bachelorproef is nagaan als elastic stack een juiste keuze is voor deze casus.

Deze bachelorproef zal een antwoord zoeken voor enkele onderzoeksvragen die werden opgesteld aan de hand van de casus.

- Kan het opzetten en gebruiken van elastic stack zonder voorkennis op een relatief korte termijn aangeleerd worden?
- Kan elastic stack autonoom werken na de initialisatiefase?
- Wat zijn op korte en lange termijn de gevolgen voor de hardware als elatic stack geïmplementeerd wordt?
- Bezit elastic stack de mogelijkheid anomalieën te detecteren?
- Is het mogelijk live duidelijk alerts te genereren?
- Hoe goed scoort deze oplossing voor de beschreven casus?

1.3 Literatuurstudie

Elastic stack is een open source project waarvan alle broncode terug te vinden is op github. Elastic stack bestaat uit drie core elementen: Logstash, Elasticsearch en Kibana. Deze drie elementen werden samengebracht tot Elastic stack en werken nu op een erg vlotte manier samen. Elastic stack heeft nu ook tools en deze zijn samen gebracht in X-Pack. Er zijn vier tools die besproken zullen worden, namelijk: Watcher, Shield, Monitoring en Machine Learning. Deze zijn gekozen omdat ze potentieel mogelijkheden hebben met de data die gegenereerd zullen worden uit de log files. Elk van deze programma's is geschreven in java en beschikt over uitgebreide documentatie (Elastic, 2017a). Binnen de Elastic stack wordt dan weer gebruik gemaakt van zelf ontwikkelde talen. Ook deze beschikken over de nodige documentatie die via de Elastic site te verkrijgen is. Op dit moment zijn met uitzonderingen geen stukken voorbeeldcode te vinden. De oorzaak hiervan is dat de elastic stack nog volop aan het groeien is. Dit is te danken aan de populariteit van Elastic stack die snel toeneemt. Zo is er een sprong gemaakt van versie 3 naar versie 5 waar heel wat basisfunctionaliteiten gewijzigd zijn. In het verloop van deze bachelorproef zal gewerkt worden met versie 5.2 .

Verder is als voor het schrijven van deze bachelorproef ook gebruik gemaakt van de boeken: (Turnbull, 2013) en (Turnbull, 2014). Ook de bachelorproef van Lintacker (2016) hielp bij de kennismaking met Elastic stack.

Bedrijven als eBay, Netflix, The Guardian, ... gebruiken elastic stack voor het beheren van hun data (Scott, 2016). Dit wijst er dus op dat het geschikt is om te werken met terabytes data. Dit zijn bedrijven die hun code liever niet delen. Er is dus niet geweten in welke mate ze gebruik maken van elastic stack en welke uitbreidingen zij eventueel gemaakt hebben.

Aangezien de voorbeelden online beperkt zijn, zal de grootste bron van informatie de documentatie van Elastic zelf zijn. Ook het aantal boeken is eerder beperkt. Het grootste probleem is dat door de snelle evolutie van elastic stack heel wat documentatie outdated is, zeker na de sprong van versie 3 naar 5. Gelukkig is er wel een actief forum[1] waarop je meestal antwoord krijgt binnen de paar uur.

[1] https://discuss.elastic.co/

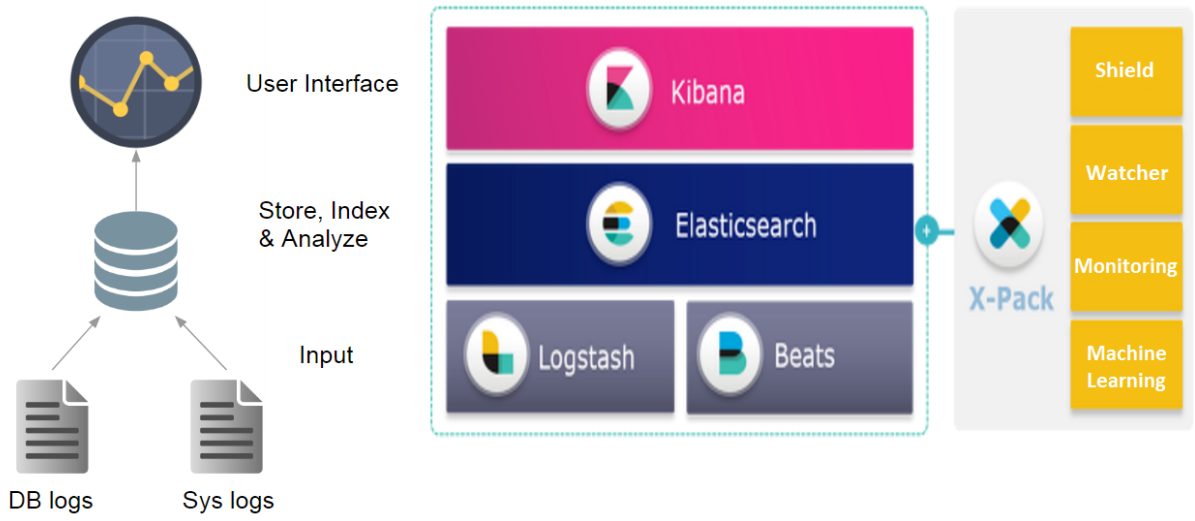
1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

Eerst en vooral zal de casus geschetst worden in 3. Daarna zal in 2 aan de hand van een overzicht de globale werking van Elastic stack uitgelegd worden. In de daarop volgende hoofdstukken 4, 5 en 6 wordt de werking van elke core component van elastic stack toegelicht. Er zal ook telkens onmiddelijk gekeken worden welke mogelijkheden dit met zich meebrengt voor onze casus. 7 bevat de evalutie van alle onderzoeksvragen die uit de casus gehaald werden. In Hoofdstuk 8, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de vraag als elastic stack een meerwaarde kan hebben voor oXya. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein. Ook zal de mening van Oxya gevraagd worden omtrent de bruikbaarheid van elastic stack voor monitoring. De installatie zal niet aan bod komen omdat deze reeds heel goed uitgelegd is op de elastic downloadpagina.

2. Elastic stack

Hier zal getracht worden aan de hand van 2.1 de werking van Elastic stack uit te leggen. Dit zal nog heel oppervlakkig gebeuren maar zou het mogelijk moeten maken een beter inzicht te creëren in de komende hoofdstukken.



Figuur 2.1: Overzicht Elastic stack

Zoals te zien is in 2.1 bestaat Elastic stack uit drie niveaus. Om de werking uit te leggen moet onderaan begonnen worden. Op dit niveau staan Logstash en Beats. Beats staat in voor het verzamelen van alle log files. Het is een tool die alle logs kan verzenden naar een centrale server waarop Logstash zich dan bevindt. Logstash leest log files en verwerkt alle nieuwe lijnen die in de file geschreven worden. Het neemt lijn per lijn en verwerkt deze tot een JSON-element.

Een niveau hoger is Elasticsearch te vinden. Elasticsearch staat in voor het opslaan en verwerken van alle input die het krijgt van Logstash. Het krijgt de JSON-elementen die Logstash aanmaakt en slaat deze zo op in zijn databank. Doordat het JSON-elementen zijn, werkt de manier van zoeken op een andere dan met SQL-opdrachten. In het verder verloop van deze bachelorproef zal gesproken worden over elementen, dit zijn dan JSON-elementen, die zich in de database bevinden.

Kibana is de grafische kant van het hele gebeuren. Hier kan een overzicht van alle elementen getoond worden, het uitvoeren van Elasticsearch queries, het maken van grafieken, ... Eenmaal alles geïmplementeerd is, zal (in normale omstandigheden) alle acties via Kibana verlopen.

Helemaal rechts kan dan nog X-Pack gevonden worden. Dit is een samenraapsel van tools die werken met Elastic stack.

3. Casus

Zoals steeds is oXya opzoek naar nieuwe manieren om nog beter te kunnen monitoren. Een aanvulling op hun monitoringtools zou dus het gebruiken van log files kunnen zijn. Deze files worden enkel bekeken als er zich een probleem voordeed om uit te zoeken wat misgelopen is. Bij oXya ontstond zo de vraag als het niet mogelijk was om deze files te gaan monitoren en zo sneller problemen te zien komen. Na wat research vonden ze dat Elastic stack in opmars is. Op het eerste gezicht heeft het de functionaliteiten waarnaar ze opzoek zijn. De concrete vraag is dus het ontwikkelen van een monitoringtool die gebruik maakt van Elastic stack om hun log files te analyseren. In de volgende alinea's zal beschreven worden wat belangrijk is voor oXya.

Bij oXya is het gebruikelijk dat elke werknemer kennis heeft van alles. Ze verwachten dus dat deze zowel kennis heeft van Linux, Windows, SAP, oracle, ... Dus om de werknemers niet nog extra te belasten, is het belangrijk dat het gebruik van de tool makkelijk aan te leren is. Een tool die makkelijk is in gebruik zal dan ook vaker gebruikt worden. Deze bachelorproef zal dan ook zo geschreven worden dat deze gebruikt kan worden voor het aanleren van enkele basismogelijkheden met Elastic stack.

Het is ook belangrijk dat de tool autonoom kan werken. Het is dus niet de bedoeling dat iemand op een knop moet drukken om te zien als er meldingen zijn. Maar wel dat deze zich automatisch live meldt als ze zich voordoen.

De implementatie mag ook maar beperkte negatieve gevolgen hebben voor de klant. Zo kan niet verwacht worden dat de klant zijn servers moet upgraden. Er zal dus bekeken worden wat de impact is voor een klant zodat oXya de afweging kan maken.

Oxya werkt met allerlei servers, platformen en databases, die elk hun eigen log files en alertfiles hebben en deze hebben ook elk hun eigen formaat of record layout. Vermits dit

een groot volume aan data kan vertegenwoordigen, moet er ook gefilterd worden in de data. Het moet dus mogelijk zijn om alles te normaliseren en dan samen te brengen tot één geheel. Ook het uitbreiden moet mogelijk zijn indien ze in de toekomst een nieuw programma willen gebruiken.

Verwacht wordt dat het programma een probleem ziet aankomen (Goldberg & Shan, 2015). Dit kan door het herkennen van bepaalde errorcodes in de logs of door een stijging in een bepaald aantal errors of het aantal logs. Dus elke afwijking van het normaal gedrag, ook wel anomalie genaamd, moet gemeld worden. Enkele mogelijkheden hiervoor zullen dus bekeken worden en geëvalueerd.

Als een probleem zich zou voordoen, wordt verwacht dat er een duidelijk alert opgesteld wordt. Het alert moet weergeven wat het probleem juist is en wat de reden is. Aangezien een klant over meerdere servers beschikt, is het ook belangrijk aan te geven op welke server het probleem zich juist bevindt. Het is ook de bedoeling dat er geen overbodige alerts gegenereed worden, want anders zal geen aandacht meer besteed worden aan deze alerts.

4. Logstash

4.1 Introductie

Logstash is een open source programma dat data verzamelt. Het werkt realtime en zal dus reageren van zodra er nieuwe data beschikbaar zijn. Het heeft een vijftigtal input mogelijkheden zoals: databanken, grafieken, files, live webpagina's, ... In deze bachelorproef zal slechts gebruik gemaakt worden van één optie, namelijk files, meer specifiek log files. Dus als verder in deze bachelorproef gesproken wordt over data dan gaat dit strikt over de data die in log files gevonden kan worden. Logstash beschikt over de mogelijkheid de data te normaliseren. Normaliseren is het omzetten van iets naar een standaardformaat. Met logstash zal dus gezorgd worden dat vergelijkbare elementen gemaakt zullen worden waar later meegewerkt kan worden. Logstash kan vanuit verschillende bronnen tegelijk inlezen en verwerken. De kracht van Logstash zit hem erin dat het realtime werkt. Dit is zeer belangrijk voor de opzet van deze bachelorproef aangezien het de bedoeling is dat er onmiddellijk een alert ontstaat van zodra iets misloopt. Het beschikt ook over enkele functies om de data te verwerken. Voorbeelden hiervan zijn: mix, match,filter, ... Het beschikt eveneens over meer dan 200 plugins en een wereldwijde community. Het doel van Logstash is voornamelijk het normaliseren van data. Dit is belangrijk voor later in Elasticsearch. Elke lijn uit een log file zal gematcht worden aan een patroon. Dit patroon zal zo enkele vaste indeces vastleggen waar later mee gewerkt zal worden. Al deze regels worden vastgelegd in een config file. Per type log zal dus een nieuwe config file geschreven moeten worden. Voor deze bachelorproef zal het beperkt worden tot de syslogs van een Linux machine en de logs van een Oracle database. Logstash werkt in drie fases: input, filter, output. Bij input zal meegegeven worden welke files juist in de gaten gehouden zullen worden. Bij de filter zullen de data verwerkt worden, dit gebeurt lijn per lijn. Elke lijn wordt beschouwd als een nieuw element. In de filter gebeuren de nodige bewerkingen

op een lijn om zo te matchen met enkele indices. Deze indices zullen dan samengebracht worden en vormen per lijn een JSON-element. In het output deel wordt beslist wat met de JSON-elementen moet gebeuren. Dit is enkel een korte beschrijving van de drie delen, maar ze beschikken over nog enkele mogelijkheden. Enkele van deze extra mogelijkheden zullen nog aan bod komen wat verder in deze bachelorproef. Logstash draait standaard op poort 9600. Maar van zodra meer dan één config file gebruikt wordt, stijgt het poort nummer telkens met 1.

4.2 Config files

Voor elk type logs moet een config file ontwikkeld worden. Deze config files moeten aan een vast patroon voldoen. Dit vast patroon bestaat uit drie delen en deze zullen hieronder uitgediept worden. Logstash beschikt reeds over heel wat functionaliteiten en deze kunnen hier niet allemaal besproken worden. Er zal beperkt worden tot de meest relevante functionaliteiten met betrekking tot dit onderzoek. Een config file voldoet aan de normale filename conventies en heeft de extensie ".conf". Logstash heeft ook enkele voor geprogrammeerde patronen. Deze patronen zijn kunnen gebruikt worden binnen de volledige config file (Elastic, 2014).

De vier config files geschreven in functie van deze bachelorproef zijn terug te vinden in A.

4.3 Input

In 4.3.1 en 4.3.2 zal besproken worden hoe we files kunnen inlezen. Een combinatie van input streams is mogelijk. Deze files worden normaal lijn per lijn gelezen en verwerkt. Indien dit niet gewenst is, kan dit aangepast worden naar wens. Dit zal besproken worden in 4.3.3.

4.3.1 Lokaal

Het simpelste is als een file lokaal te vinden is. Dit houdt dus in dat de file op dezelfde (virtuele) machine te vinden is als waarop de Logstash geïnstalleerd werd. Er dient dan wel gezorgd te worden voor een account die read rechten heeft voor deze file. Als meerdere files aan hetzelfde format voldoen kunnen deze allemaal samen ingelezen worden. Het is belangrijk dat het absolut path gegeven wordt.

```
input {
        file {
                path => "/var/log/messages"
                path => "/var/log/auth.log"
                path => "/var/log/kern.log"
                path => "/var/log/cron.log"
        }
}
```

4.3.2 Beats

Beats is een programma dat nu ook tot de Elastic stack behoort. Het wordt niet beschouwd als core maar eerder als tool. Het is een licht programma dat nieuwe data uit een file realtime via het netwerk naar een andere machine kan sturen. Dit gebeurt via een open netwerkpoort op die andere machine waarop Logstash staat. Logstash is op hetzelfde moment aan het luisteren naar die poort en kan zo de nieuwe data verkrijgen en dan ook verwerken. Ook moet voor de configuratie van de beats het ip-adres en de open poort van de andere machine gekend zijn. In het volgend voorbeeld zal hiervoor de nog niet gebruikte poort 5043 gebruikt worden. En het ip-adres is 192.168.0.100.

```
input {
        beats {
                port => "5043"
        }
}
```

Dit is het stuk code die in de config file hoort op de machine waarop Logstash staat.

```
filebeat.prospectors:
- input_type: log
        paths:
                - /var/log/messages

output.logstash:
        hosts: ["192.168.0.100:5043"]
```

Dit is het stuk code die in de Beats config file hoort .

4.3.3 Multiline

Multiline is een plugin en moet dus nog geïnstalleerd worden als deze voor de eerste maal gebruikt wordt. Als u zich in de folder van Logstash bevindt, is het commando hiervoor:

```
bin/logstash-plugin install logstash-filter-multiline
```

Multiline kan ervoor zorgen dat een file niet lijn per lijn gelezen wordt, maar dat hij bepaalde lijnen samen inleest. Dit is nodig voor onze Oracle logs, soms bestaat één message uit meerdere lijnen. Er kan een patroon opgegeven worden waaraan lijnen wel of net niet moeten voldoen om afzonderlijk te kunnen bestaan. Er kan meegegeven worden wat er moet gebeuren met een lijn die niet aan de wensen voldoet. Als deze aan de lijn ervoor of erna toegevoegd moet worden. Een probleem dat opdook was dat Logstash nooit de laatste lijn las. Dit was omdat lijnen die niet alleen konden bestaan aan de lijn ervoor toegevoegd werden. Logstash kon dus pas een message inlezen als hij zeker was dat ze volledig was. Omdat de log message altijd in één keer geschreven wordt, kan ervanuit gegaan worden dat de message compleet is. Daarom werd een auto-flush geïmplementeerd die als er binnen de seconde geen nieuwe lijn komt de message als compleet zal beschouwen.

```
input {
        ...
        codec => multiline {
                pattern => "%{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{
                    YEAR}"
                negate => true
                what => "previous"
                auto_flush_interval => 1
        }
}
```

In pattern wordt het gewenste patroon geplaatst. Dit door middel van een regex of de voorgeprogrammeerde patronen. In negate wordt aangegeven als het patroon wel of net niet moet matchen. De what geeft hier aan dat de lijnen die niet voldoen aan het patroon aan de vorige lijn toegevoegd zullen worden. En de auto-flush staat op één seconde om de delay te beperkten.

4.3.4 Windows eventlogs

Windows vergt een andere manier van werken voor het verkrijgen van de eventlogs. Om deze te verkrijgen hoeft geen pad opgegeven te worden, maar wel het type en welke eventlog gewenst is.

```
input {
  eventlog {
    type  => 'Win32-EventLog'
    logfile  => 'Application'
  }
}
```

4.4 Filter

Hier gebeuren alle bewerkingen en zal bepaald worden wat in de JSON gaat en wat niet. In de filter is het de bedoeling dat de file zoveel mogelijk genormaliseerd wordt. Een log lijn wordt eerst volledig ingelezen in een field genaamd “message”. Van daaruit is het de bedoeling dat vast voorkomende stukken tekst eruit gehaald worden en in een eigen veld geplaatst worden, indien deze later gebruikt willen worden. Om deze specifieke velden te maken zijn enkele mogelijkheden. Deze zullen besproken worden in 4.4.1 en 4.4.2. Werken met de tijd is niet altijd even simpel en zal daarom afzonderlijk besproken worden in 4.4.3. Er zal afgesloten worden in 4.4.4 met nog enkele nuttige commando’s. Hou er rekening mee dat Elasticsearch zal zoeken aan de hand van de fields die hier gemaakt worden. De juiste field keuzes kunnen dus een grote invloed hebben op de zoeksnelheid.

4.4.1 Patroon

Om een file te normaliseren zal gebruik gemaakt worden van een patroon. Aan de hand van dit patroon worden enkele fields gemaakt. Belangrijk is dus dat waarden in een log steeds op dezelfde plaats te vinden zijn. Indien dit het geval is, kan hier het meeste werk van de config file gebeuren. Hiervoor zal een regex opgesteld worden die gebruik maakt van de voorgeprogrammeerde patronen. Met %{TYPE:name} kan een field name gemaakt worden die voldoet aan het voorgeprogrammeerde type. Op het einde van de logs die in deze casus gebruikt worden, staat op het einde van de lijn een boodschap. Om deze boodschap ook in een field te krijgen werd gebruik gemaakt van een tweede mogelijkheid. Met (?<name>regex) kan weer een field name gemaakt worden die voldoet aan de regex binnen de haakjes.

```
filter {
        grok {
                match {
                        "message" => "%{MONTH:month} +%{MONTHDAY:
                           monthday} %{TIME:time} %{NOTSPACE:user
                           }(?<log_message>.*$)"
                }
        }
}
```

4.4.2 Zoeken

Sommige programma's hebben een eigen type errors of messages. Deze message zijn opgebouwd volgens een vast patroon. Dit maakt het mogelijk om met logstash te zoeken naar dit patroon en op die manier deze messages in een field te steken. De error messages van oracle kunnen hier als voorbeeld dienen. Deze message zijn opgebouwd uit "ORA-" gevolgd door vijf cijfers. Er zal dus gezocht worden als dit patroon in de message te vinden is en als dat het geval is, zal deze in een field geplaatst worden.

```
filter {
        if [message] =~ /ORA-/ {
                grok {
                        match => [ "message","(?<ORA->ORA-[0-9]*)"
                           ]
                }
        }
}
```

4.4.3 Timestamp

Logstash zal altijd als standaard timestamp het moment nemen waarop hij de data verwerkt. Als dit live gelezen wordt, is dit goed maar als er ook logs van vroeger ingeladen worden, geeft dit problemen. Daarom zal meestal de tijdsaanduiding in het begin van een log lijn gebruikt worden als timestamp. Als bijvoorbeeld het jaar niet gegeven werd, zal logstash automatisch het jaar nemen waarin het zich bevindt. Er wordt nu vanuit gegaan dat de

tijdsaanduiding gecapteerd kan worden aan de hand van het patroon dat uitgelegd werd in 4.4.1. Eerst wordt een tijdelijk field gemaakt met alleen de tijdsaanduidingen. Daarna kan de tijdzone gekozen worden en de timestamp gelijk gezet worden met ons tijdelijk field. Er moet wel meegegeven worden hoe het tijdelijk field opgebouwd is.

```
filter {
        ...
        mutate {
                add_field => {
                        "timestamp" => "%{year} %{month} %{monthday
                           } %{time}"
                }
        }

        date {
                timezone => "CET"
                match => [ "timestamp","yyyy MMM dd HH:mm:ss" ]
        }
}
```

4.4.4 Overige

Enkele nuttige commando's die nog niet aan bod kwamen zijn het kopiëren en het verwijderen van fields. Zo kan het voorbeeld handig zijn om message op het einde te vervangen door de log message alleen zonder alle andere info. Deze info zit dan toch al in andere fields en zo kan de message makkelijker leesbaar gemaakt worden. Fields die gemaakt werden voor tijdelijk gebruik kunnen best verwijderd worden. Dit om de simpele reden dat ze enkel extra geheugen in nemen en toch tot niets dienen.

4.5 Output

Voor de output zijn weer heel wat mogelijkheden. In deze bachelorproef zullen er slechts drie aan bod komen. Hier zal dus beschreven worden wat er moet gebeuren met de JSON-elementen.

4.5.1 Standaard output

Dit is vooral nuttig voor tijdens de testfase. Als logstash dan gestart wordt vanuit de commandline kan de output live gevolgd worden. Dit heeft als voordeel dat onmiddellijk gezien kan worden als het werkt zonder errors. De code hiervoor is dan ook heel simpel.

```
output {
        stdout {
                codec => rubydebug
        }
}
```

4.5.2 Elasticsearch

Door het samenbrengen van deze programma's binnen de Elastic stack is het zeer makkelijk om de elementen door te geven. Zo kan in enkele lijnen duidelijk gemaakt worden dat alle JSON-elementen naar Elasticsearch moeten. Om Elasticsearch efficiënt te laten werken wordt een index verwacht. Deze index wordt dan toegevoegd aan elk JSON-element die hier gegenereerd werd. Aangeraden wordt om in eerste instantie het type logs te vermelden in de index voorbeeld: Linux, Oracle, Windows, ... Als extra kan ook een combinatie gemaakt worden met de servernaam om zo duidelijk te maken waar de logs vandaan komen. Via hosts kan duidelijk gemaakt worden waar elasticsearch juist aan het draaien is. In dit voorbeeld draait elasticsearch lokaal maar er kan ook een ip adres opgegeven worden.

```
output {
        elasticsearch {
                hosts => ["localhost:9200"]
                index => "linux_citts"
        }
}
```

4.5.3 Alerts

Het is ook mogelijk alerting te doen binnen logstash. In deze bachelorproef zal, om dit te illustreren, gealert worden op belangrijke ORA-errors (orACsd, 2011). Als één van de errors voorkomt wordt deze in het field "ORA-"geplaatst (zie stap 4.4.2). Dit voorbeeld zal kijken als er zich geen invalid state voordoet. De gekende errors voor invalide state zijn: ORA-01502 en ORA-20000. Als dus één van deze errors zich voordoet zal een e-mail met een alert gestuurd worden. In de body wordt enige duiding gegeven rond waar en wanneer de error zich voordeed.

```
output {
        if "ORA-01502" == [ORA-] or "ORA-20000" == [ORA-] {
                email{
                        subject => "Invalid State"
                        to => "bvervaele@oxya.com"
                        body => "Host: %{host}\n\nTime: %{
                           @timestamp}\n\nLine of the error: %{
                           message}"
                        address => "smtp.gmail.com"
                        port => 587
                        username => "logserviceoxya@gmail.com"
                        password => "oxya1234"
                        use_tls => true
                }
        }
}
```

5. Elasticsearch en X-Pack

Elasticsearch is het hart van elastic stack. Het is de databank die alle JSON-elementen bevat die reeds gemaakt werden door Logstash. Het bevat een geavanceerd zoekalgoritme gebaseerd op indices. Voor een hoeveelheid data onder duizend elementen wordt beter iets anders gebruikt aangezien de start tijd van Elasticsearch enkele seconden bedraagt. Maar voor enkel miljoenen elementen is het zeer geschikt. Na de starttijd kan Elasticsearch zo goed als live de zoekopdrachten volbrengen. Omwille van de uitermate goede schaalbaarheid wordt het reeds bij enkele bedrijven gebruikt die met heel veel data af te rekenen krijgen (Scott, 2016).

Om Elasticsearch optimaal te laten werken, is het zeer belangrijk om juist om te gaan met indices. Als voorbeeld gezocht wil worden op een bepaalde dag van de week dan kan hiervoor best een afzonderlijk field voorzien worden. Daarom is het dus erg belangrijk om werk te maken van het schrijven van een goede config file voor logstash.

Het gebruik van de zelf ontwikkelde taal is niet eenvoudig en daarom zal in 5.1 enkele basis mogelijkheden toegelicht worden. De commando's kunnen via de console in Kibana gerund worden of via de commandline. In de rest van de bachelorproef zullen de stukken code gebruikt worden voor in de console van Kibana. Indien gewenst kunnen deze stukken code ook gebruikt worden in de commandline, maar dan wel in combinatie met curl.

X-Pack is een recente uitbreiding op de elastic stack. Deze tools groeien erg snel en er komen er nog steeds bij. Deze tools zijn gebasseerd op het sterk algoritme van Elasticsearch.

5.1 Search

Elasticsearch is eerst en vooral een database en daar horen natuurlijk zoekopdrachten bij. Elasticsearch werkt niet met sql queries, maar heeft daar een eigen taal voor. Er zijn twee soorten zoekopdrachten:

- **Leaf query:** dit zoekt voor specifieke waarden in bepaalde fields, dit door queries zoals match, term, range. Deze queries kunnen afzonderlijk gebruikt worden.
- **Compound query:** deze queries combineren leaf of andere compound queries op een logische manier (zoals met bool of dis_max).

In onderstaand stuk code worden leaf queries samengebracht tot één compound query. Eerst en vooral zal een filter uitgevoerd worden die ervoor zorgt dat de rest alleen uitgevoerd zal worden op een beperkte selectie elementen. In dit voorbeeld zal enkel gewerkt worden op elementen die backup als user hebben. Als tweede deel zal gezocht worden naar een element dat "drop table"bevat. Deze zoekopdrachten werden uitvoerd op de elementen met de index linux.

```
POST /linux
{
    "query": {
        "bool" : {
            "must" : {
                "query_string" : {
                    "query" : "drop table",
                }
            },
            "filter" : {
                "term" : { "user" : "backup" }
            }
        }
    }
}
```

Er zijn nog heel wat mogelijkheden en deze zijn terug te vinden in de documentatie. Zoekopdrachten kunnen nuttig zijn in het zoeken naar een probleem. Maar voor het monitoren is dit minder van toepassing. De toepassingen die als volgende besproken worden, maken hier in de achtergrond natuurlijk wel gebruik van.

5.2 Watcher

Watcher is één van de tools die tot de Elastic stack behoren. Het kijkt uit naar veranderingen of anomalieën in de data en voert de nodige vervolgacties uit (Elastic, 2017b). Er zijn enkele belangrijke eigenschappen nodig om watcher te kunnen gebruiken:

- de relavante data of data verandering kan periodiek opgevraagd worden met een query,
- de resultaten van de query kunnen gematcht worden aan de voorwaarden,

- één of meerdere acties worden ondernomen als de voorwaarden voldaan zijn.

Een Watch query bestaat uit vier delen: planning, zoekopdracht, conditie en actie.

- **Planning:** een planning voor het laten lopen van een zoekopdracht en het checken van de conditie.
- **Zoekopdrach:** de zoekopdracht is de input voor de conditie. Het ondersteunt de volledige elasticsearch.
- **Conditie:** is een conditie die beslist wanneer er acties moeten volgen. Het is mogelijk hier te werken met true or false maar ook het schrijven van meerdere scenarios is toegestaan.
- **Actie:** een e-mail zenden, 3rd party software waarschuwen, de opgevraagde data stockeren, ...

Dit is dus ook een erg belangrijke tool voor het monitoren. Watcher zal gebruikt worden om duidelijke boodschappen te e-mailen van waar zich een probleem voor doet. Dit is ook de tool die gebruikt moet worden om alerts te voorzien bij het gebruik van grafieken of andere tools.

5.3 Shield

Shield maakt het mogelijk op een eenvoudige manier de Elastic stack te beveiligen. Met het gebruik van Shield wordt alle data pas vrijgegeven na het inloggen met gebruikersnaam en paswoord. Dit zorgt ervoor dat niemand met slechte bedoelingen zo maar aan de data zou kunnen. Maar tegenwoordig zijn er al andere manieren om die data toch te kunnen bemachtigen. Daarom zijn er ook meer geavanceerde mogelijkheden zoals:

- data-encryptie voor communicatie tussen de nodes,
- beperkte toegang voor bepaalde users,
- IP-filtering.

In shield zit ook een module die zorgt voor data integriteit. Het bevat een eigen protocol dat de data zal encrypten tussen de nodes en daarna nog een authenticatie bericht zal zenden. Dit zorgt er voor dat er geen foute data of een tekort aan data zal ontstaan.

Indien shield geïmplementeerd wordt, dienen ook de config files in logstash aangepast worden. Deze moeten dan voorzien worden van een gebruikersnaam en paswoord.

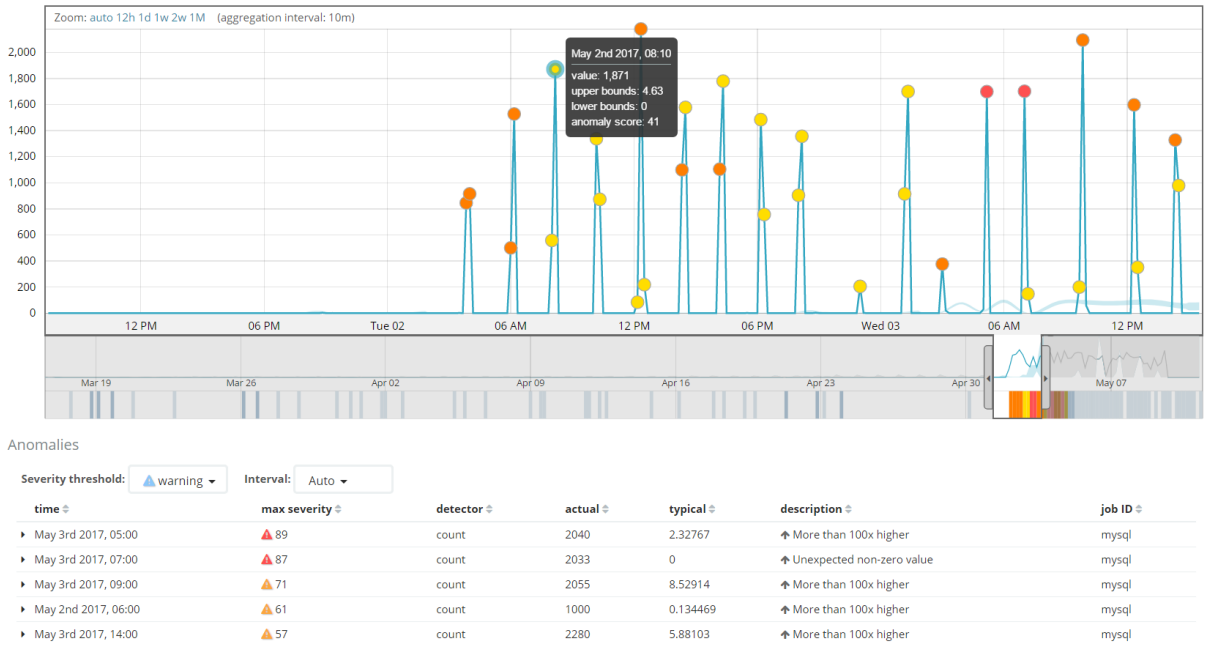
5.4 Monitoring

Deze monitoring tool maakt het mogelijk om via Kibana de volledige Elastic stack te monitoren. Onder dit monitoren wordt verstaan het bekijken van de preformance van de Elastic stack in realtime. Er wordt een agent geïnstalleerd die uit alle nodes metrtics gaat verzamelen. Deze data worden dan verwerkt en zal aan de hand van een dashboard

de prestaties van de Elastic stack weergeven. Het kijkt naar de prestaties zoals hoelang zoekopdrachten duren in Elasticsearch. Hier wordt ook nog systeeminformatie gegeven. Dingen zoals geheugen gebruik, aantal gebruikte shards, uptime, . . . (Simpson, 2013).

5.5 Machine learning

Deze tool kwam uit in 5.4 en was nog in beta maar werd met zicht op de casus toch opgenomen (Chang & Dodson, 2017). Bij Machine Learning draait het er om dat de machine slimmer wordt en gaat herkennen wanneer zich een anomalie voor doet. Hoe meer input data hoe beter het onregelmatigheden zal gaan herkennen. Het berekent een beneden- en bovenlimiet. Afhankelijk van de afwijking daarvan zal een anomalie score berekend worden. Deze scores worden dan nog eens in vier categorieën opgedeeld. Dan kan gekozen worden op welk van de vier categorieën gealert wordt en op welke niet.



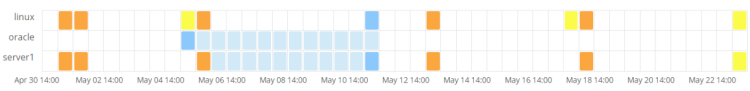
Anomalies

Severity threshold: warning Interval: Auto

time	max severity	detector	actual	typical	description	job ID
May 3rd 2017, 05:00	89	count	2040	2.32767	More than 100x higher	mysql
May 3rd 2017, 07:00	87	count	2033	0	Unexpected non-zero value	mysql
May 3rd 2017, 09:00	71	count	2055	8.52914	More than 100x higher	mysql
May 2nd 2017, 06:00	61	count	1000	0.134469	More than 100x higher	mysql
May 3rd 2017, 14:00	57	count	2280	5.88103	More than 100x higher	mysql

Figuur 5.1: Machine Learning toegepast op de hoeveelheid loglijnen gecreëerd door een MSSQL.

Er kan ook een vergelijking gemaakt worden tussen verschillende reeksen binnen Machine Learning om te zien als er een verband bestaat tussen hen. Dit kan ervoor zorgen dat de bron van een probleem sneller gevonden kan worden.



Figuur 5.2: Overzicht van anomalieën bij: Server

6. Kibana

6.1 Introductie

Kibana is het programma dat voor de visualisatie binnen de Elastic stack zorgt. Deze zal default te vinden zijn op poort 5601. De eerste keer dat Kabina gestart wordt, zal gevraagd worden voor een index pattern. Kibana kan dus pas gebruikt worden als er al data in Elasticsearch zit. Aan de hand van het index pattern kan al gefilterd worden. Het is ook mogelijk om een regex op te geven als index pattern. "*"zal alle data inlezen en gebruiken. Als "Time-field name"zal in deze casus altijd "@timespamp"gebruikt worden. Deze is in logstash zo ingesteld om overeen te komen met de timestamp van de log zelf.

Eenmaal deze instelling gebeurd is, kan Kibana in gebruik genomen worden. Aan de linkerkant bevinden zich enkele symbolen die alle opties weergeven. De opties die voor de casus van belang zijn, zullen hieronder verduidelijkt worden.

6.2 Discover

Hier kunnen alle JSON-elementen gevonden worden die matchen met het opgegeven index pattern. Bovenaan wordt een grafiek getoond van wanneer hoeveel logs gegenereerd zijn. Aan de linkerkant wordt dan weer een lijst getoond met welke fields allemaal te vinden zijn binnen deze selectie van data. De voorstelling van de data ziet er chaotisch uit op het eerste gezicht. Deze kan eenvoudig gewijzigd worden door bij een element op het pijltje te drukken zodat alle fields voor dit element getoond worden. Deze fields kunnen dan gebruikt worden in de tabel. Om een field als kolom vast te zetten in de tabel moet éénmaal gedrukt worden op het derde icoontje. Zo kan dus een overzichtelijke tabel bekomen en

inzicht in de data verkregen worden.

6.3 Visualize

Onder visualize valt een grote variatie aan dingen: grafieken, getallen, tekst, heatmap, ... Voor deze casus zal voornamelijk gebruik gemaakt worden van grafieken en getallen. Het is belangrijk om de juist keuze te maken van visualisatie, zowel van de keuze voor de visualisatie als de data die gebruikt worden ervoor.

Als gekozen wordt om een grafiek te maken, is het belangrijk eerst te kiezen wat er op de assen moet komen. Indien gewenst kan deze grafiek nog verder onderverdeeld/gefilterd worden door sub-buckets toe te voegen. Deze onderverdeling kan bijvoorbeeld gemaakt worden op de value van een field. Dan dient binnen de sub-bucket gekozen te worden voor Terms en dan het gewenste field. Ook een filter wordt via sub-buckets toegevoegd. Een filter juist definiëren wordt besproken in 6.6.

Er is ook nog de mogelijkheid tot het schrijven van een script om de grafiek nog meer naar wens te maken (Bragin, 2016).

6.4 Timelion

In timelion kunnen grafieken met elkaar vergeleken worden. Zo kan na gegaan worden hoe een grafiek er de vorige weken uitzag ten op zichte van die van deze week. Voor de casus kan dit zeker van pas komen voor het ontdekken van abnormale afwijkingen.

Er zijn reeds heel wat mogelijkheden geïmplementeerd (Elastic, 2016). Maar om nuttige grafieken te verkrijgen zal veelal een combinatie gemaakt moeten worden van enkele van deze mogelijkheden.

Helaas kan hier nog geen gemiddelde en standaard afwijking berekend worden van dezelfde momenten in de afgelopen weken. Dit zou voor deze casus handiger zijn omdat er 's morgens pieken zullen zijn die op die manier niet voorspeld kunnen worden.

6.5 Dashboard

In dashboard kunnen alle visualisaties samengebracht worden. Hier kan dus een duidelijk overzicht van de data gecreëerd worden. De plaatsing en keuze van grafieken zijn natuurlijk wel van belang. Om het overzichtelijk te houden kunnen afzonderlijke dashboards gemaakt worden. Zo kan voor deze casus voorbeeld een dashboard gemaakt worden voor SAP, Oracle, Windows, ... De dashboards zijn dan ook een belangrijke informatiebron bij het zoeken naar een fout. Hier kan de zoektoch gestart worden om te kijken als er geen opvallende afwijkingen waren of om een idee te krijgen van wanneer iets fout liep.

6.6 Filters

Er is reeds een filter gebeurd bij het kiezen van het index pattern. Maar er zijn nog heel wat mogelijkheden. De filter die meest gebruikt zal worden, is een filter op de tijd. De tijd kan rechts bovenaan gewijzigd worden. Zo kan er naar een specifiek moment in de tijd gegaan worden om te zien wat er gebeurd is.

Voor de rest van het filteren wordt je aangewezen op de zoekbalk bovenaan. Er kan simpelweg gezocht worden op een woord dat voorkomt in een JSON-element. Zo kan bijvoorbeeld gezocht worden op een specifieke ORA-error door in de zoekbalk de code in te geven. In de zoekbalk kunnen ook combinaties van filters gebeuren.

Er is ook de mogelijkheid om te zoeken op de waarde van een field. Dit kan aan de hand van volgende query: fieldName:"waarde".

Niet elk JSON-element heeft alle fields. Als men enkel de elementen wil zien met een field genaamd "code"kan gezocht worden op volgende query: *_exists_* : ”*code*”.

Als iets net niet mag voorkomen in een zoekopdracht kan men er een "!"voor typen.

7. Evaluatie casus

Hier zal bekeken worden als Elastic stack voldoet aan de vooropgestelde eisen in de casus. Zowel de voor- als nadelen zullen aan bod komen en enkele eventuele alternatieven. Het is de bedoeling een duidelijk beeld te schetsen van de sterktes en zwaktes van Elastic stack. Deze evaluatie gebeurde samen met Dhr. Van Den Abeele van oXya.

7.1 Nodige kennis

Om dit te testen werd een werknemer van het bedrijf Oxya gevraagd enkele handelingen uit te voeren. Hij kreeg daarvoor de hoofdstukken Logstash, Elasticsearch en X-Pack en Kibana ter beschikking en de mogelijkheid om via het internet te zoeken. Er werd ook telkens op voorhand een schatting gemaakt van de benodigde tijd. De werknemer werd het volgende gevraagd:

1. schrijf een config file voor logstash voor het lezen van een Linux syslog (2h),
2. zoek in Elasticsearch op hoeveel logs er het laatste uur waren met het woord ”sap” in de message (30min),
3. maak een grafiek die het aantal logs met een ”word” field per tijd terug geeft (10min),
4. intreperteer uit een gegeven dashboard wanneer iets is fout gelopen (2min).

1. De werknemer nam eerst snel het hoofsdstuk Logstash eens door. Daarna werd gekeken hoe hij de file kon normaliseren. Eenmaal dat beslist was, ging hij aan de slag met de input. Aangezien het path gegeven was, was de input correct geschreven op 2 minuten. Aangezien op voorhand goed nagedacht was hoe de normalisering moest gebeuren, ging het schrijven van het grok pattern ook vlot. Als output werd std out gebruikt en Elasticsearch. Dit stond goed beschreven en was snel correct geschreven. Bij het eerste keer runnen van de config

file werd gezien dat de datum nog niet juist was en er ook nog fields te veel waren. Het juist krijgen van de datum vergde nog wat extra tijd. Het verwijderen van de overbodige fields was dan weer heel snel gebeurd.

Om een volledig werkende config file te maken, zonder externe hulp, heeft het 1 uur en 6 minuten geduurd.

2. Ook hier ging de werknemer niet direct over tot de actie, maar nam eerst zijn tijd om bijpassend hoofdstuk door te nemen. Daarna heeft hij twee afzonderlijke queries geschreven die afzonderlijk bleken te werken na enkele pogingen. Na was prutsen met alle haakjes werden de twee queries samengebracht en dit gaf het gewenste resultaat in net geen 23 minuten.

3. Toen de werknemer niet direct een sectie zag over een grafiek maken, ging deze direct over naar de praktijk. Aangezien de symbolen duidelijk zijn, werd op geen tijd gevonden waar hij de juiste grafiek kon maken. Een grafiek die het volledige aantal logs per tijd weergaf, ontstond in minder dan 5 minuten. Daarna kwam de moeilijkheid om enkel de logs te tonen met het ”word” field. De werknemer werd nog enkele minuten gegeven waarin hij verschillende settings aanpaste. Na twaalf minuten was de gewenste grafiek nog niet in orde en werd de opdracht gestopt.

4. De werknemer kon na wat inzoomen op de grafieken tot op de minuut zeggen wanneer het was fout gelopen. Hij zei ook nog dat deze opdracht wel erg makkelijk was.

Hieruit wordt besloten dat het aanleren van de Elastic stack vlot verliep. Zeker de meest gebruikte stap, het gebruik van het dashboard, werd aangegeven makkelijk te zijn. Vermeld moet wel worden dat gebruik gemaakt werd van enkele goed gekozen grafieken die problemen snel terug geven. Aangezien het maken van een speciale grafiek niet zo eenvoudig bleek te zijn werd een extra sectie geschreven.

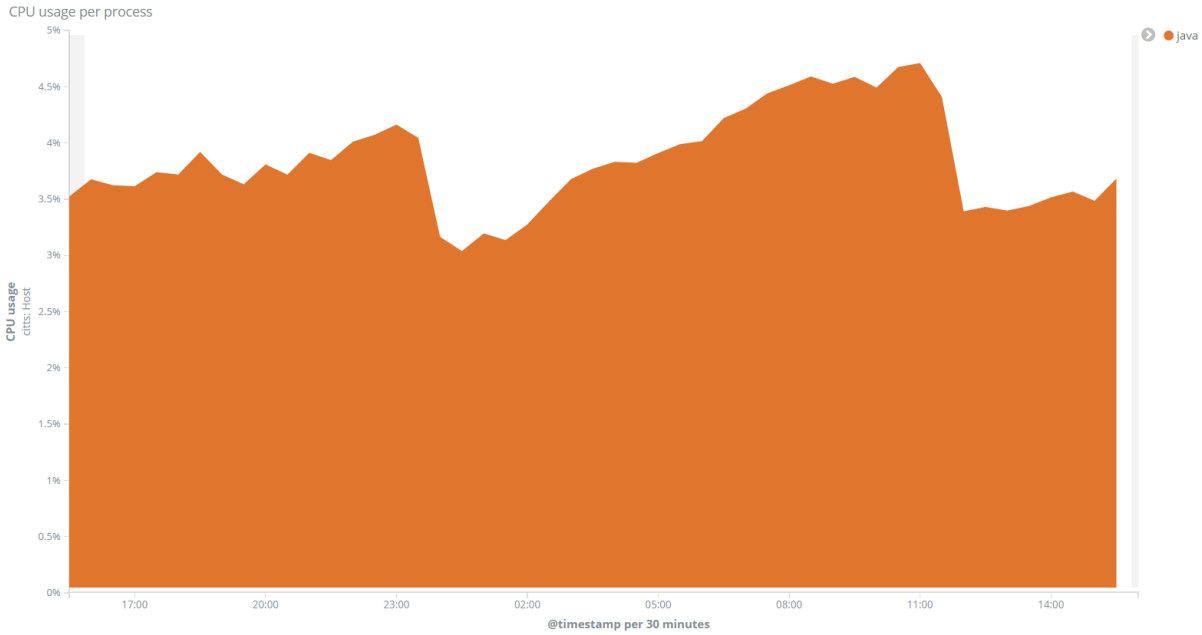
7.2 Autonomie

De Elastic stack kan autonoom werken en zal alerts genereren zonder dat daarvoor iets extra hoeft gedaan te worden. Het is natuurlijk wel aan de eindgebruiker om te zorgen dat de grafieken en alerts relevant blijven. Dit zal dus nu en dan wat tijd vergen, maar dit zal eerder beperkt zijn. Na de ontwikkeling kan gesteld worden dat Elastic stack autonoom zal werken en een probleem live kan gaan alerten.

7.3 Hardware vereisten

Belangrijk om mee te beginnen is dat na de installatie geen restart nodig is. Restarts worden zoveel mogelijk vermeden omdat de klant dan al zijn processen moet stoppen en dit voor overlast zorgt.

Aangezien deze bachelorproef slechts een proof of concept is, werd alles dan ook enkel geïmplementeerd op een testserver. Op deze testserver kon wel het cpu gebruik geobserveerd en geëvalueerd worden. Er bestaat een programma, Metricbeat genaamd, dat systeeminformatie opvraagt. Metricbeat kan op enkele minuten tijd met Elastic stack verbonden worden. Online kan een default dashboard gevonden worden die de systeem info weergeeft. Dit leek een heel eenvoudige manier om uit te vinden hoeveel cpu elk onderdeel gebruikte. Helaas valt alles onder de noemer java en wordt dan ook zo getoond. In 7.1 werden alle Java processen samen gemonitord voor 24 uur.



Figuur 7.1: CPU gebruik van Java over 24 uur.

Om dus te weten hoeveel nu echt naar de Elastic stack gaat, moet ook het cpu gebruik voor java gemeten worden als Elastic stack af staat. Als Elastic stack uitgezet wordt, kan deze monitoring niet gebruikt worden. Daarom werden enkele steekproeven uitgevoerd en bleek dat Java dan gemiddeld 1.7% van de CPU gebruikt. Dus de Elastic stack gebruikt gemiddeld 2% van de cpu op de testserver. De testserver is dan ook nog eens veel lichter dan de servers bij de klanten. Dit zorgt ervoor dat een klant amper een verschil in CPU gebruik zal waarnemen door het implementeren van een Elastic stack.

Voor de disk space werden enkele berekeningen uitgevoerd. Als basis werd het gebruikte geheugen en het aantal elementen bekeken. Om dit te vinden gebruikten we volgend commando in de commandline:

```
curl -XGET 'http://localhost:9200/_nodes/stats'
```

Hieruit werd gevonden dat de testomgeving reeds 3502 elementen in de Elasticsearch heeft opgeslagen. Deze elementen zouden 1.968.548 bytes innemen. Dit zou willen zeggen dat één JSON-lement dus ongeveer 562 bytes in beslag neemt. De testomgeving bevat een voldoende groot aantal en variatie aan elementen om deze gegevens als representatief te beschouwen voor verdere berekeningen. Daarna werd uitgezocht hoeveel elementen grote

bedrijven als Stapels en Bekaert zouden hebben op één maand. Om dit te doen werden het aantal loglijnen geteld en gekeken over welke periode deze geproduceerd werden. Indien de log slechts de laatste 24 uur bijhield werd een gemiddelde van drie verschillende dagen genomen.

Staples (Linux)

- syslog: 10.603 elementen/dag
- oracle: 280.322 elementen/40 dagen

Bekaer (Windows)

- syslog: 314.044 elementen/270 dagen
- MySQL: 30.891 elementen/175 dagen

Voor Staples zou dit betekenen dat er elke maand 528.331 elementen bijkomen. Dit komt overeen met 296.922.022 bytes of 297MB. Bij bekaert zou een maand slechts 40.190 elementen bevatten. Dit komt dan weer overeen met 22.586.780 bytes of 22.5MB.

Er kan dus besloten worden dat, voor het beperkt aantal log files die voor deze casus bijgehouden worden, de nodige hoeveelheid disk space erg beperkt blijft. Ook valt op dat een windowsserver nog minder aan disk space zal in nemen dan een linuxserver.

7.4 Flexiebel

oXya heeft geen vaste pakketten. Het maakt aanbiedingen op maat van elke klant. Hierdoor zitten ze dus met verschillende besturingssytemen en databanken. Als besturingssystemen werden redhat en windows server 2012R2 getest. En als databases werden oracle en MySQL bekeken. De config files zijn terug te vinden in A. Deze config files kunnen dus voor elk type logfile geschreven worden. Er is wel een verschil in hoe ver een file genormaliseerd kan worden. Een oracle file zal voorbeeld verder genormaliseerd kunnen worden dan een MySQL file. Dit zal voor MySQL de monitoringmogelijkheden beperken. Maar er kan dus gesteld worden dat het flexibel genoeg is aangezien het eender welke log files kan gebruiken als input data.

7.5 Annomaly detectie

oXya zou graag op de hoogte gebracht worden als er zich ergens uitzonderlijk gedrag voor doet. De mogelijkheid is er om statische thresholds te plaatsen. Maar in plaats daarvan wordt gekozen om te werken met dynamische thresholds. Dit om te verkomen dat bij een groeiend bedrijf telkens de waarden aangepast moeten worden. In 7.5.1 zal bekeken worden wat de mogelijkheden zijn met de vrijheid die daar gegeven wordt. In 7.5.2 daarentegen is de vrijheid beperkt, maar indien het van genoeg input voorzien wordt, is het zeker betrouwbaar.

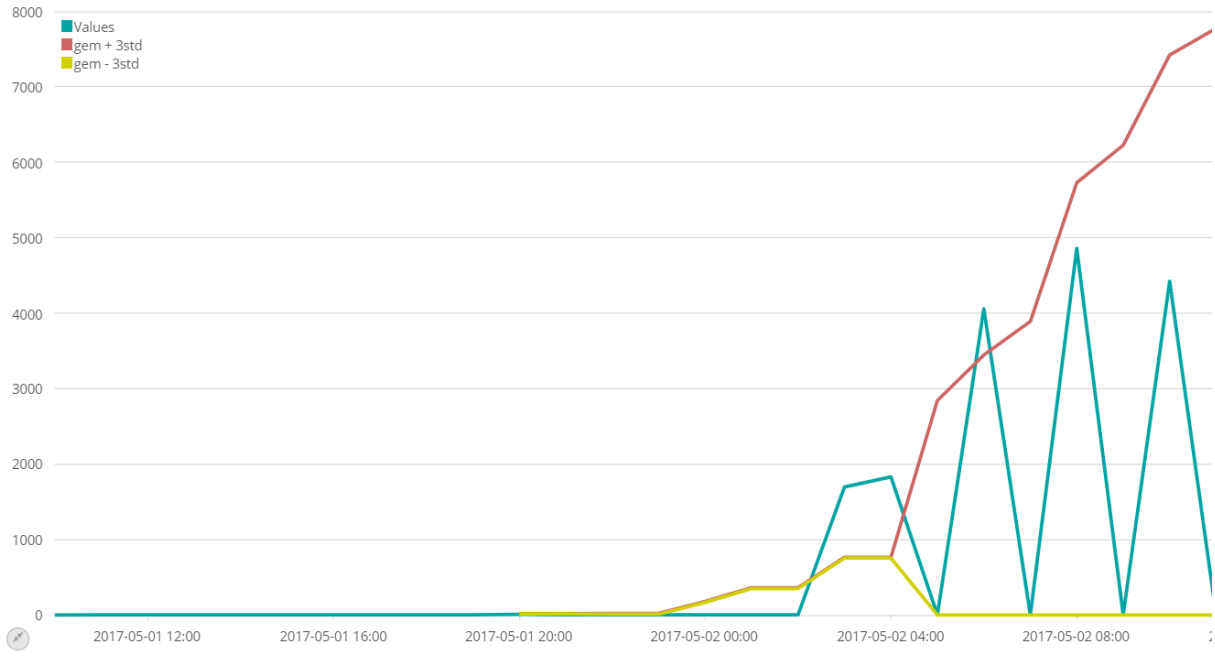
De servers die oXya monitort zijn meer belast in de week dan in het weekend. Ook overdag is er meer activiteit dan 's nachts. Het ideale zou dus zijn om te gaan vergelijken met hetzelfde moment in de voorbije weken (Mohan, 2017).

Over het algemeen beschikt Elastic stack dus over enkele nuttige anomaliedetectie mogelijkheden. Er is een breed aanbod en afhankelijk van de toepassing zal dus een gepaste anomaliedetectie gekozen moeten worden.

7.5.1 Timelion

Eén van de mogelijkheden is om de gegevens van dit moment te gaan vergelijken met het gemiddelde $\pm$ 3 keer de standaard deviatie van de voorbije waarden. Dit zou willen zeggen dat met een 99,7% zekerheid gezegd zou kunnen worden als een waarde uitzonderlijk is of niet. Om het gemiddelde en de standaard deviatie te berekenen zullen de waarden van de laatste 24 uur gebruikt worden. In deze situatie worden beste resultaten verkregen als de waarden per uur gegroepeerd worden. Om deze grafiek te creëren dienen volgende commando's uitgevoerd te worden:

```
        .es(index="mssql").label("Values"),
        .es(index="mssql",offset=-1h).movingaverage(10).max(15).sum
            (.es(index="mssql",offset=-1h).movingstd(10).multiply(3)
            ).label("gem + 3std"),
        .es(index="mssql",offset=-1h).movingaverage(10).max(15).sum
            (.es(index="mssql",offset=-1h).movingstd(10).multiply
            (-3)).label("gem - 3std")
```



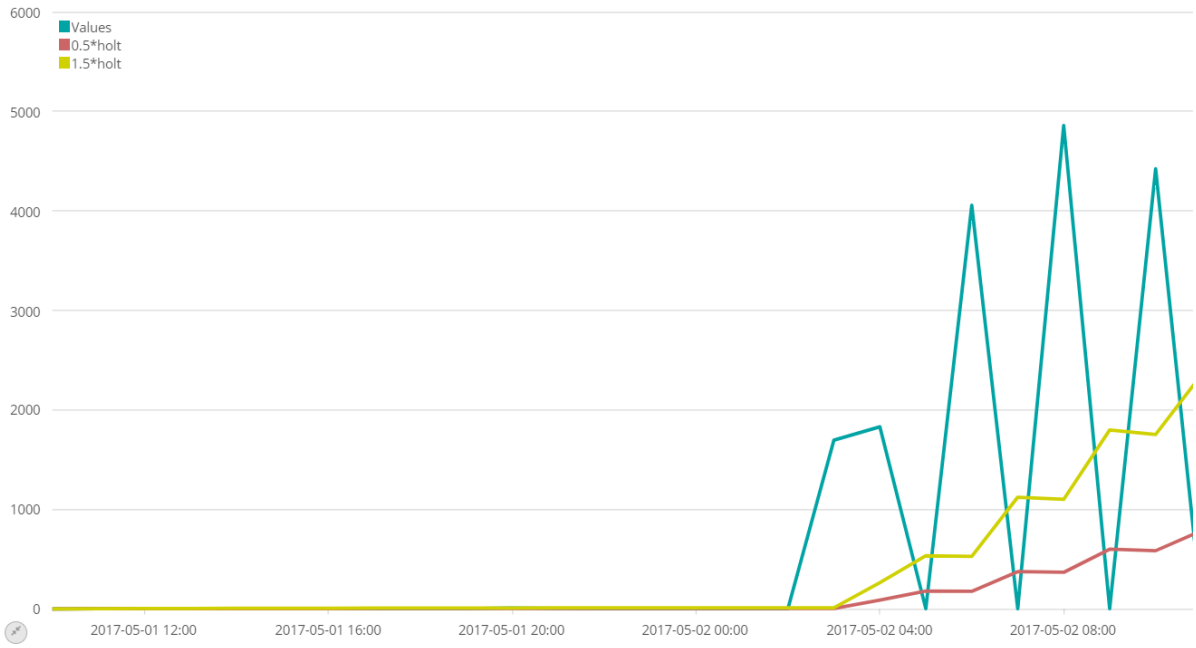
Figuur 7.2: Anomaliedetectie aan de hand van het gemiddelde en de standaard deviatie.

De grafiek van dit moment moet zich dus tussen de andere twee grafieken bevinden anders

dient er een alert opgesteld te worden voor een anomaly.

Een andere mogelijkheid met timelion is de implementatie van het holt-winter algoritme (Goodwin, 2010). Ook dit heeft zijn voor- en nadelen. Het zal meer rekening gaan houden met de trend van de grafiek. Het enige probleem hiermee is dat er nog geen betrouwbaarheidsintervalen geïmplementeerd zijn. Wat wel een mogelijkheid zou zijn is dat de grafiek zich tussen 0,5 en 1,5 keer de verwachte holt-winter waarde moet bevinden.

```
        .es(index="mssql").label("Values"),
        .es(index="mssql",offset=-1h).holt(0.1,0.1).multiply(0.5).
            label("0.5*holt"),
        .es(index="mssql",offset=-1h).holt(0.1,0.1).multiply(1.5).
            label("1.5*holt")
```



Figuur 7.3: Anomalie detectie aan de hand van Holt Winters.

Maar ook hier is het weer mogelijk om andere dingen mee te doen door een combinatie te maken met andere mogelijkheden. Hier zijn dus heel wat mogelijkheden en de uitkomst zal afhangen van het gekozen algoritme.

7.5.2 Machine Learning

Zoals reeds besproken is er bij Machine Learning, naast het kiezen uit de vier warnings, geen vrijheid. Maar dit hoeft ook niet, want als de juiste elementen hier als input gebruikt worden doet Machine Learning de rest. Hoe meer input Machine Learning krijgt hoe accurater er een alert gestuurd kan worden. Deze tool is nog in beta en heeft dus ook enkele kinderziektes. Zo werd ontdekt dat na het ontstaan van een vijftal anomanlieën op een termijn van 24 uur Machine Learning het moeilijk heeft om terug naar een stabiele

toestand te gaan. Zo zou het in de daaropvolgende week mogelijk zijn dat nog grotere anomalieën niet opgemerkt worden.

Machine Learning heeft dus nog enkele minpunten, maar deze wegen niet op tegen de toch al nuttige functionaliteit.

7.5.3 Eigen detectie

Indien bovenstaande tools nog niet aan de wensen voldoen, is er nog altijd de optie om een eigen detectie tool te ontwerpen. Het ontwikkelen van een eigen detectie tool is mogelijk door gebruik te maken van de java api (Tyson, 2016). Bij het ontwikkelen van een eigen detectie tool in java kan dan gebruik gemaakt worden van alle Elasticsearch queries. De java api zorgt ervoor dat de limiet van anomalie detectie bij jezelf ligt.

7.6 Alers

De alerts die in Logstash gegenereed worden bevatten in de titel het probleem en op welke host het probleem zich stelt. Als de e-mail dan geopend wordt, kan men terug vinden wanneer het probleem zich voor deed. Nog eens op welke machine het probleem vastgesteld werd. En dan de boodschap die bij het probleem hoort. Deze boodschap is voor een werknemer van oxya voldoende om te kunnen beslissen als er opvolging nodig is of niet.

Ook de alerting via Watcher kan duidelijk verlopen. Watcher biedt ook een breed gamma aan mogelijkheden aan. Door oXya werd gekozen om de alerting via email te laten verlopen. Watcher maakt het mogelijk om te alerten op Elasticsearch queries. Maar Watcher kan ook gebruikt worden om te alerten op Timelion of op de Machine Learning.

Dus ook op gebied van alerting zijn er genoeg mogelijkheden.

8. Conclusie

Elastic stack voldoet aan alle vereisten vooropgesteld door oXya. Indien beslist zou worden om er mee verder te gaan, zal een analyse gemaakt moeten worden van welke andere log files nog gebruikt kunnen worden. Voor deze files zal dan een Logstash config file geschreven moeten worden. Dit zorgt er voor dat de implementatie wel wat werk zal vergen, maar daarna werkt het volledig autonoom. Elastic stack zou er dan voor zorgen dat sommige problemen vroegtijdig gemeld kunnen worden. Sommige tools en het dashboard kunnen dan geraadpleegd worden bij het onderzoeken van het probleem.

Naar de toekomst toe zijn dus heel wat mogelijkheden. Elastic stack is ook nog steeds erg snel aan het groeien en zal dus nog meer mogelijkheden bieden in de toekomst. Voor oXya is dit, volgens de geschetste casus, een gepast product. Maar de uitdaging zal liggen in het op de gepaste tijd zenden van meldingen. Een mogelijk vervolg onderzoek zou kunnen zijn naar de anomaly detectie. Een andere mogelijkheid zou zijn naar het zelf schrijven van elastic aangezien het open scource is en een java api heeft.

Hieronder wordt de mening van oXya zelf gegeven:

“De eerste resultaten bij het gebruik van Elastic stack om een bijkomende vorm van monitoring te hebben, lijken zeer positief. De mogelijkheid om verschillenden log files van uiteenlopende systemen (os, db , sap) te analyseren en daarin bepaalde kritische alerts te detecteren en te escaleren is zeer bruikbaar in onze omgeving. Het online oppikken van alerts was een optie die te verwachten was van deze tool. Maar wat nog interessanter blijkt te zijn, is het gedrag van bepaalde loglijnen in log files te laten analyseren en te laten interpreteren met de “Timelion” en de “Machine Learning“ modules. Wat de mogelijkheid geeft om statische beslissingen te treffen, tijdens het analyseren van deze (soms seasonal) data. Bijvoorbeeld het gebruik van de "Seasonal Holt-Winters"methode om abnormaal

gedrag te voorspellen, lijken duidelijk een stap in de goede richting. Waar we vroeger enkel triggers gebruikten om alerts te generen, kunnen we nu ook anomalieën vroegtijdig detecteren. Gezien er nog veel mogelijkheden te onderzoeken zijn in het domein van machine learning zijn we zeker van plan nog verder onderzoek in te plannen op dit terrein."

-Van Den Abeele, Martin

Bibliografie

Bragin, T. (2016). Using Painless in Kibana scripted fields. Verkregen 13 december 2016, van https://www.elastic.co/blog/using-painless-kibana-scripted-fields

Chang, S. & Dodson, S. (2017). Machine Learning in the Elastic Stack. Verkregen 8 maart 2017, van https://www.elastic.co/elasticon/conf/2017/sf/machine-learning-in-the-elastic-stack?baymax=rtp&elektra=videos&storm=video3&iesrc=ctr

Elastic. (2014). grok-patterns. Verkregen 10 april 2014, van https://github.com/elastic/logstash/blob/v1.4.2/patterns/grok-patterns

Elastic. (2016). Timelion function reference. Verkregen 18 april 2017, van https://github.com/elastic/timelion/blob/master/FUNCTIONS.md

Elastic. (2017a). Elastic Stack and Product Documentation. Verkregen 18 april 2017, van https://www.elastic.co/guide/index.html

Elastic. (2017b). Watcher Lab — Creating Alerts with Dynamic Threshold. Verkregen 15 april 2017, van https://www.elastic.co/videos/watcher-lab-creating-alerts-with-dynamic-threshold

Goldberg, D. & Shan, Y. (2015). The Importance of Features for Statistical Anomaly Detection. Verkregen 18 april 2017, van https://www.usenix.org/system/files/conference/hotcloud15/hotcloud15-goldberg.pdf

Goodwin, P. (2010). The Holt-Winters Approach to Exponential Smoothing: 50 Years Old and Going Strong. *FORESIGHT*, *19*, 30–33.

Lintacker, K. (2016). *Lead-in tot data analyse op logfiles met behulp van de ELK stack* (Bachelorproef, Hogeschool Gent).

Mohan, V. (2017). Elasticsearch Aggregations: Weekday and Hourly Analysis. Verkregen 3 januari 2017, van https://qbox.io/blog/elasticsearch-aggregations-weekday-hourly-analysis

orACsd. (2011). ORACLE Consolidation Security Availability Performance on Unix Systems (and a start on MS SQL). Verkregen 22 maart 2011, van http://oracsd.blogspot.be/2011/03/alert-log-file-critical-errors-list.html

Scott, S. (2016). These 15 Tech Companies Chose the ELK Stack Over Proprietary Logging Software. Verkregen 17 februari 2016, van https://logz.io/blog/15-tech-companies-chose-elk-stack/

Simpson, C. (2013). Elasticsearch IJellow"cluster status explained. Verkregen 1 april 2013, van http://chrissimpson.co.uk/elasticsearch-yellow-cluster-status-explained.htmll

Turnbull, J. (2013). *The Logstash Book*. lulu.com.

Turnbull, J. (2014). *The art of monitoring*. James Turnbull.

Tyson, M. (2016). Use Elasticsearch in your Java applications. Verkregen 8 maart 2016, van https://www.ibm.com/developerworks/library/j-use-elasticsearch-java-apps/index.html

A. Logstash config files

Linux:

```
input {
    file {
        path => "/var/log/messages"
        path => "/var/log/auth.log"
        path => "/var/log/kern.log"
        path => "/var/log/cron.log"
    }
}

filter {

    grok {
        match => {
            "message" => "%{MONTH:month} +%{MONTHDAY:monthday} %{
                TIME:time} %{NOTSPACE:user}(?<log_message>.*$)"
        }
    }

    if [message] =~ /SAPFDG_00/ {
        grok {
            match => [ "message","(?<tcode>[A-Z]([A-Z]|[0-9])[0-9])
                " ]
        }
    }
    if [message] =~ /dump/ {
        grok {
            match => [ "message","(?<word>dump)" ]
        }
    }
    if [message] =~ /semaphore/ {
```

```
        grok {
            match => [ "message","(?<word>semaphore)" ]
        }
    }
    if [message] =~ /LOAD_NO_ROLL/ {
        grok {
            match => [ "message","(?<word>LOAD_NO_ROLL)" ]
        }
    }
    if [message] =~ /lock/ {
        grok {
            match => [ "message","(?<word>lock)" ]
        }
    }
    if [message] =~ /storage/ {
        grok {
            match => [ "message","(?<word>storage)" ]
        }
    }
    if [message] =~ /authentication failure/ {
        grok {
            match => [ "message","(?<word>authentication failure)"
               ]
        }
    }
    if [message] =~ /memory bottleneck/ {
        grok {
            match => [ "message","(?<word>memory bottleneck)" ]
        }
    }
    if [message] =~ /runtime error/ {
        grok {
            match => [ "message","(?<word>runtime error)" ]
        }
    }
    if [message] =~ /snapshot/ {
        grok {
            match => [ "message","(?<word>snapshot)" ]
        }
    }


    mutate {
        add_field => {"[hour]" => "%{+HH}"}
        add_field => {"[weekday]" => "%{+EEE}"}
    }

    date{
        locale => "en"
        match => ["timestamp","MMM d HH:mm:ss","MMM dd HH:mm:ss","
           ISO8601"]
    }

    mutate { replace => [ "message", "%{log_message}" ]  }

    mutate {
```

```
        remove_field => [ "time" ,"month","monthday","timestamp","
            log_message","@version"]
    }

\}

output {

    if "dump" in [message]{
        email{
            subject => "Dump on %{host}"
            to => "bvervaele@oxya.com"
            body => "Host: %{host}\n\nTime: %{@timestamp}\n\nLine
                of the error: %{message}"
            address => "smtp.gmail.com"
            port => 587
            username => "logserviceoxya@gmail.com"
            password => "oxya1234"
            use\_tls => true
        }
    }

    if "semaphore" in [message]{
        email{
            subject => "Semaphore on %{host}"
            to => "bvervaele@oxya.com"
            body => "Host: %{host}\n\nTime: %{@timestamp}\n\nLine
                of the error: %{message}"
            address => "smtp.gmail.com"
            port => 587
            username => "logserviceoxya@gmail.com"
            password => "oxya1234"
            use\_tls => true
        }
    }

    elasticsearch {
        hosts => ["localhost:9200"]
        user => elastic
        password => changeme
        index => "linux"
    }
    stdout {
        codec => rubydebug
    }
}
```

Windows:

```
input {
  beats {
    host=>"localhost"
    port=>"2056"
  }
}

filter {
        mutate {
        remove_field => [ "Category"
,"ComputerName","EventIdentifier","InsertionStrings","logfile","
   RecordNumber","SourceName","TimeGenerated","TimeWritten","
   EventType","RecordNumber","type"]
    }
}
output {

    elasticsearch {
         hosts => ["localhost:9200"]
        user => elastic
        password => changeme
        index => "windows"
    }
    stdout {
        codec => rubydebug
    }
}
```

Oracle:

```
input {
    file {
        path => "/oracle/FDG/saptrace/diag/rdbms/fdg/FDG/trace/
           alert_FDG.log"
        codec => multiline {
            pattern => "%{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{YEAR}
               "
            negate => true
            what => "previous"
            auto_flush_interval => 1
        }
    }
}

filter {

    if [message] =~ /Starting ORACLE instance/ {
        mutate {
            add_field => [ "oradb_status", "starting" ]
        }
    } else if [message] =~ /Instance shutdown complete/ {
        mutate {
            add_field => [ "oradb_status", "shutdown" ]
        }
    } else {
        mutate {
            add_field => [ "oradb_status", "running" ]
        }
    }

    if [message] =~ /ORA-/ {
        grok {
            match => [ "message","(?<ORA->ORA-[0-9]*)" ]
        }
    }

    if [message] =~ /current log/ {
        grok {
            match => [ "message","(?<word>current log)" ]
        }
    }
    if [message] =~ /deadlock/ {
        grok {
            match => [ "message","(?<word>deadlock)" ]
        }
    }
    if [message] =~ /DATABASE OPEN/ {
        grok {
            match => [ "message","(?<word>DATABASE OPEN)" ]
        }
    }
    if [message] =~ /SYSTEM ARCHIVE LOG/ {
        grok {
            match => [ "message","(?<word>SYSTEM ARCHIVE LOG)" ]
```

```
        }
    }
    if [message] =~ /CONTROLFILE/ {
        grok {
            match => [ "message","(?<word>CONTROLFILE)" ]
        }
    }
    if [message] =~ /FULL/ {
        grok {
            match => [ "message","(?<word>FULL)" ]
        }
    }

   mutate {
        add_field => {"[hour]" => "%{+HH}"}
        add_field => {"[weekday]" => "%{+EEE}"}
    }


    grok {
        match => [ "message","%{DAY:day} %{MONTH:month} %{MONTHDAY:
            monthday} %{TIME:time} %{YEAR:year}(?<log\_message>.*$)"
             ]
    }

    mutate {
       add_field => {
            "timestamp" => "%{year} %{month} %{monthday} %{time}"
       }
    }

    date {
        timezone => "CET"
        match => [ "timestamp" , "yyyy MMM dd HH:mm:ss" ]
    }

    mutate { replace => [ "message", "%{log_message}" ]  }

    mutate {
        remove_field => [ "time" ,"month","monthday","year","
            timestamp","day","log\_message"]
    }
}

output {
    if "ORA-00257" == [ORA-] or "ORA-16038" == [ORA-]{
    email{
            subject => "Archiver Hung on %{host}"
            to => "bvervaele@oxya.com"
            body => "Host: %{host}\n\nTime: %{@timestamp}\n\nLine
                of the error: %{message}"
            address => "smtp.gmail.com"
            port => 587
            username => "logserviceoxya@gmail.com"
            password => "oxya1234"
            use_tls => true
```

```
        }
    }

    if "ORA-01114" == [ORA-] or "ORA-01157" == [ORA-] or "ORA-01578
        " == [ORA-] or "ORA-27048" == [ORA-]{
        email{
            subject => "Data Block Corruption" on %{host}
            to => "bvervaele@oxya.com"
            body => "Host: %{host}\n\nTime: %{@timestamp}\n\nLine
                of the error: %{message}"
            address => "smtp.gmail.com"
            port => 587
            username => "logserviceoxya@gmail.com"
            password => "oxya1234"
            use_tls => true
        }
    }

    if "ORA-01243" == [ORA-]{
        email{
            subject => "Media Failure on %{host}"
            to => "bvervaele@oxya.com"
            body => "Host: %{host}\n\nTime: %{@timestamp}\n\nLine
                of the error: %{message}"
            address => "smtp.gmail.com"
            port => 587
            username => "logserviceoxya@gmail.com"
            password => "oxya1234"
            use_tls => true
        }
    }

    if "ORA-01502" == [ORA-] or "ORA-20000" == [ORA-] {
        email{
            subject => "Invalid State on %{host}"
            to => "bvervaele@oxya.com"
            body => "Host: %{host}\n\nTime: %{@timestamp}\n\nLine
                of the error: %{message}"
            address => "smtp.gmail.com"
            port => 587
            username => "logserviceoxya@gmail.com"
            password => "oxya1234"
            use_tls => true
        }
    }

    if "ORA-01652" == [ORA-] or "ORA-01653" == [ORA-] or "ORA-01654
        " == [ORA-] {
        email{
            subject => "Extension Error on %{host}"
            to => "bvervaele@oxya.com"
            body => "Host: %{host}\n\nTime: %{@timestamp}\n\nLine
                of the error: %{message}"
            address => "smtp.gmail.com"
            port => 587
            username => "logserviceoxya@gmail.com"
```

```
            password => "oxya1234"
            use_tls => true
        }
    }

    if "ORA-00600" == [ORA-] or "ORA-07445" == [ORA-] or "ORA-04"
        in [ORA-]{
        email{
            subject => "Generic Error on %{host}"
            to => "bvervaele@oxya.com"
            body => "Host: %{host}\n\nTime: %{@timestamp}\n\nLine
                of the error: %{message}"
            address => "smtp.gmail.com"
            port => 587
            username => "logserviceoxya@gmail.com"
            password => "oxya1234"
            use_tls => true
        }
    }

    if "ORA-00020" == [ORA-]{
        email{
            subject => "Maximum number of processes exceeded on %{
                host}"
            to => "bvervaele@oxya.com"
            body => "Host: %{host}\n\nTime: %{@timestamp}\n\nLine
                of the error: %{message}"
            address => "smtp.gmail.com"
            port => 587
            username => "logserviceoxya@gmail.com"
            password => "oxya1234"
            use_tls => true
        }
    }
    elasticsearch {
        hosts => ["localhost:9200"]
        user => elastic
        password => changeme
        index => "oracle"
    }
    stdout {
        codec => rubydebug
        id => "test outprint"
    }
}
```

MSSQL:

```
input {
    file {
        path => "C:\Program Files\Microsoft SQL Server\MSSQL10_50.
            MSSQLSERVER\MSSQL\Log"
                codec => multiline {
            pattern => "%{YEAR:year}-%{MONTH:month}-%{MONTHDAY:
                monthday} %{TIME:time}"
            negate => true
            what => "previous"
                        auto\_flush_interval => 1
        }
    }
}

filter {
    grok {
        match => {
            "message" => "%{YEAR:year}-%{MONTH:month}-%{MONTHDAY:
                monthday} %{TIME:time} %{USERNAME:username}\s*(?<log
                \_message>.*$)"
        }
    }

        mutate {
       add_field => {
            "timestamp" => "%{year} %{month} %{monthday} %{time}"
       }
    }

    date{
        locale => "en"
        match => ["timestamp","yyyyy mm dd HH:mm:ss","yyyyy mm dd
            HH:mm:ss","ISO8601"]
    }

    mutate { replace => [ "message", "%{log_message}" ]  }

    mutate {
        remove\_field => [ "log\_message","@version"]
    }

 hosts => ["localhost:9200"]
        user => elastic
        password => changeme
        index => "oracle"\}

output {

    if "dump" in [message]{
        email{
            subject => "Dump"
            to => "bvervaele@oxya.com"
            body => "Host: %{host}\n\nTime: %{@timestamp}\n\nLine
                of the error: %{message}"
```

```
            address => "smtp.gmail.com"
            port => 587
            username => "logserviceoxya@gmail.com"
            password => "oxya1234"
            use\_tls => true
        }
    }

    if "semaphore" in [message]{
        email{
            subject => "Semaphore"
            to => "bvervaele@oxya.com"
            body => "Host: %{host}\n\nTime: %{@timestamp}\n\nLine
                of the error: %{message}"
            address => "smtp.gmail.com"
            port => 587
            username => "logserviceoxya@gmail.com"
            password => "oxya1234"
            use\_tls => true
        }
    }

    elasticsearch {
        hosts => ["localhost:9200"]
        user => elastic
        password => changeme
        index => "mssql"
    }
    stdout {
        codec => rubydebug
    }
}
```

Lijst van figuren